

Amendments to the Claims:

This listing of claims replaces all prior versions and listings of claims in the application:

Listing of Claims:

1. (Currently Amended) A method of operating a processor, the method comprising:
executing a branch instruction in execution of an instruction stream with the branch instruction including a first token that specifies the number of instructions in the instruction stream that are after the branch instruction to execute before performing the branch operation and a second token that specifies a branch guess operation to cause the processor, if the first token specifies zero or one instructions to execute after the branch, to prefetch, prior to performing an evaluation of a branch condition associated with the branch instruction, an instruction for the “branch taken” condition rather than a next sequential instruction; and
retrieving, based, at least in part, on the second token, one of the instruction for the “branch taken” condition and the next sequential instruction.
2. (Cancelled)
3. (Previously presented) The method of claim 1 wherein the branch instruction further comprises an optional token that indicates a pipeline stage that the branch operation is evaluated in.
4. (Previously presented) The method of claim 1 wherein the first token can specify, one, two or three instructions following to execute before performing the branch operation.
5. (Cancelled).
6. (Previously presented) The method of claim 1 wherein at least one of the tokens are specified by a programmer or assembler program to enable variable cycle deferred branching.

7. (Previously presented) The method of claim 1 wherein the tokens are specified to assist an assembler program to produce more efficient code.

8. (Previously Presented) The method of claim 1 wherein the branch instruction is a branch unconditionally or branch to an instruction at a specified label based on an ALU condition code.

9. (Cancelled).

10. (Previously Presented) The method of claim 1 wherein the branch instruction is a branch instruction that causes the processor to branch to the instruction at a specified label if a specified byte in a longword matches or mismatches a `byte_compare_value`.

11. (Previously Presented) The method of claim 1 wherein the branch instruction is a branch instruction that causes the processor to branch to the instruction at a specified label based on whether or not a current context is a specified context in the branch instruction.

12. (Previously presented) The method of claim 1 wherein the branch instruction is a branch instruction that causes the processor to branch if the state of a specified state name is a selected value.

13. (Previously presented) The method of claim 1 wherein the branch instruction is a branch instruction that causes the processor to branch if a specified signal is deasserted.

14-16. (Cancelled).

17. (Currently amended) A method of operating a processor, the method comprising:
executing a branch instruction that causes a branch operation in an instruction ~~stream~~
stream;

deferring performance of the branch operation of the branch instruction based on
evaluating a first token that specifies a number of instructions to execute before performing the
branch operation; and

evaluating a second token that specifies a branch guess operation which causes the
processor, if the first token specifies zero or one instructions to execute after the branch, to
prefetch, prior to performing an evaluation of a branch condition associated with the branch
instruction, an instruction for the "branch taken" condition rather than a next sequential
instruction; and

retrieving, based, at least in part, on the evaluated second token, one of the instruction for
the "branch taken" condition and the next sequential instruction.

18. (Cancelled)

19. (Previously presented) The method of claim 17 wherein the first token and the
second token are selectable by a programmer.

20. (Previously presented) The method of claim 17 wherein the first token and the
second token are specified by a programmer or assembler program to enable variable cycle
deferred branching.

21. (Previously presented) The method of claim 17 wherein at least one of the first
token and the second token is specified to assist an assembler program to produce more efficient
code.

22. (Currently amended) A processor comprising:
decode logic, including an instruction decoder and a program counter, for decoding
instructions, the decode logic ~~including logic~~ configured to:

execute a branch instruction in execution of an instruction stream in the processor with the branch instruction including a first token that specifies the number of instructions in the instruction stream to execute before performing the branch operation, and a second token that specifies a branch guess operation to cause the processor, if the first token specifies zero or one instructions to execute after the branch, to prefetch, prior to performing an evaluation of a branch condition associated with the branch instruction, an instruction for the "branch taken" condition rather than a next sequential instruction; and

retrieve, based, at least in part, on the second token, one of the instruction for the "branch taken" condition and the next sequential instruction.

23. (Cancelled)

24. (Previously presented) The method of claim 1, wherein the processor is a hardware-based multithreaded processor having multiple engines to process multiple threads and the branch instruction is part of an instruction set for the multiple engines; and executing comprises:

executing the branch on one of the multiple engines.

25. (Previously presented) The method of claim 17, wherein the processor is a hardware-based multithreaded processor having multiple engines to process multiple threads and the branch instruction is part of an instruction set for the multiple engines; and executing and deferring occurs on one of the multiple engines.

26. (Previously presented) The processor of claim 22, wherein the branch instruction is part of an instruction set for a hardware-based multithreaded processor having multiple engines to process multiple threads.